Perspectives in Science (2016) 8, 159—161

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/pisc

Improved simple optimization (SOPT) algorithm for unconstrained non-linear optimization problems☆

J. Thomas[a,*], S.S. Mahapatra[b]

[a] Disha Institute of Management and Technology, Raipur, Chhattisgarh, India
[b] National Institute of Technology, Rourkela, Odisha, India

Received 6 January 2016; accepted 6 April 2016
Available online 23 April 2016

KEYWORDS
Simple optimization;
Benchmark functions;
Unconstrained optimization

Summary In the recent years, population based meta-heuristic are developed to solve non-linear optimization problems. These problems are difficult to solve using traditional methods. Simple optimization (SOPT) algorithm is one of the simple and efficient meta-heuristic techniques to solve the non-linear optimization problems. In this paper, SOPT is compared with some of the well-known meta-heuristic techniques viz. Artificial Bee Colony algorithm (ABC), Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and Differential Evolutions (DE). For comparison, SOPT algorithm is coded in MATLAB and 25 standard test functions for unconstrained optimization having different characteristics are run for 30 times each. The results of experiments are compared with previously reported results of other algorithms. Promising and comparable results are obtained for most of the test problems. To improve the performance of SOPT, an improvement in the algorithm is proposed which helps it to come out of local optima when algorithm gets trapped in it. In almost all the test problems, improved SOPT is able to get the actual solution at least once in 30 runs.

Introduction

Optimization in its simplest form refers to obtain an optimum (minimum or maximum) value of a real function by systematically selecting the real or integer variable values from within a prescribed set. If the function is linear then there are traditional methods like simplex method available to solve the problem efficiently and exactly. Solving nonlinear optimization function is more complicated in nature and many times traditional methods based on direct search and gradient based search do not able to solve the problems. Meta-heuristic algorithms are now being used by the researchers to overcome the drawbacks of the traditional methods. Though meta-heuristic algorithms do not guarantee exact solutions still gives better solutions compared to traditional techniques in most of the problems.

☆ This article belongs to the special issue on Engineering and Material Sciences.

* Corresponding author. Tel.: +91 7415328771.
E-mail address: joji.siji@gmail.com (J. Thomas).

http://dx.doi.org/10.1016/j.pisc.2016.04.021

In many complex real world problems, they become the only choice for solution.

Metaheuristic algorithms

Meta-heuristic algorithms are population based algorithms which starts with a set of randomly generated solutions and these solutions are changed in each iteration (generation) by applying some defined equations. It is observed that after some generations, the optimal or near optimal solution gets evolved from the original random generated solutions.

Genetic algorithm (GA) (Goldberg, 1989; Deb, 2004) was one of the first evolutionary algorithms in which fitness value of randomly generated population (solutions) is first calculated and then selection, crossover and mutation operators are applied to population in each generation. Fitness value is the measure of quality of the solution, higher fitness value refers to better solution. Differential evaluation (DE) (Karaboga and Akay, 2009) algorithm also uses operators similar to GA. The operators used in DE are: mutation, crossover and selection. DE relies more on mutation operator for generating new better solutions whereas GA relies on crossover operation. In this algorithm, each new solution is compared with a mutated solution and the better one is selected for the future generation. Particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) also starts with a set of randomly generated solutions which are called particles. These particles move in the search space on the basis of the position of best particle and its own best position attained so far. A best particle is one whose fitness value is maximum. In artificial bee colony algorithm (ABC) (Karaboga and Akay, 2009) randomly generated solutions are called food sources and fitness value of each solution is called nectar amount of food source. Each iteration of the algorithm completes in three phases: employed bees phase, onlooker bees phase and scout bees phase. Number of food sources, employed bees and onlooker bees are kept same.

Simple optimization (SOPT) algorithm

SOPT consists of two stages — exploration stage and exploitation stage (Hasancebi and Azad, 2012). During exploration stage, the best solution is moved according to Eq. (1) and the new solution generated. If fitness value of new solution is better than fitness value of worst solution of the population then the worst solution is replaced by the new generated solution. A similar Eq. (2) is used in exploitation stage with a difference only in the constant used in the equation is taken half of that used in Eq. (1). Again worst solution of the population is replaced with new solution if its fitness value is less than the new solution. These two stages are continued till a termination criterion is not met; in this paper it is number of function evaluations.

$$x_{i,new} = x_{i,best} + c_1 \times R_i \tag{1}$$

$$x_{i,new} = x_{i,best} + c_2 \times R_i \tag{2}$$

where $x_{i,new}$ is the ith parameter of the new solution in a particular iteration, $x_{i,best}$ is the ith parameter of the best solution in the same iteration, c_1 is a positive constant and R_i is a normally distributed random number with a mean zero and a standard deviation σ_i. σ_i is the standard deviation of ith parameters of all the members of the population.

To improve the algorithm, the best solution is replaced with a new randomly generated solution and the next best solution is used for guiding the new solutions. This replacement is made when the best solution remain unaltered for certain number of iterations. The number of iteration after which the best solution to be replaced is calculated by Eq. (3). It depends on the number of variables in the problem and population size of the algorithm.

$$\text{Replacement\_counter} = 0.5 \times N \times D \tag{3}$$

Experiments

To compare the performance of SOPT with other metaheuristic algorithms, 25 standard test functions with different characteristics are selected from the previous work of Karaboga and Akay (2009). Selected test functions for experiment are given in Table 1.

SOPT algorithm is coded using MATLAB and each function is run for 30 times independently with different seeds of random numbers. In these experiments value of c_1 is taken as 1.5 and c_2 is 0.75 that is half of c_1. Results are compared with

Table 1 Standard test functions used for experiments D, dimension; C, characteristics; U, unimodal; M, multimodal; S, separable; N, non-separable.

No	Function	D	C	Range	No	Function	D	C	Range
1	Easom	2	UN	[−100,100]	14	CamelBack	2	MN	[−5,5]
2	Matyas	2	UN	[−10,10]	15	Colville	4	UN	[−10,10]
3	Quartic	30	US	[−1.28,1.28]	16	DixonPrice	30	UN	[−10,10]
4	sphere	30	US	[−100,100]	17	Michalewicz10	10	MS	[0,π]
5	StepInt	5	US	[−5.12,5.12]	18	Michalewicz2	2	MS	[0,π]
6	Step	30	US	[−100,100]	19	Michalewicz5	5	MS	[0,π]
7	SumSquares	30	US	[−10,10]	20	Rastrigin	30	MS	[−5.12,5.12]
8	Trid10	10	UN	$[-D^2,D^2]$	21	Rosenbrock	30	UN	[−30,30]
9	Trid6	6	UN	$[-D^2,D^2]$	22	Schwefel	30	MS	[−500,500]
10	Zakharov	10	UN	[−5,10]	23	Akley	30	MN	[−32,32]
11	Bohchevsky1	2	MS	[−100,100]	24	Griewank	30	MN	[−600,600]
12	Bohchevsky2	2	MN	[−100,100]	25	Powell	24	UN	[−4,5]
13	Bohchevsky3	2	MN	[−100,100]					

Table 2 Mean function values obtained over 30 independent run.

No	Function	Actual	PSO	ABC	GA	DE	SOPT
1	Easom	−1	−1	−1	−1	−1	−1
2	Matyas	0	0	0	0	0	0
3	Quartic	0	0.00116	0.03002	0.1807	0.00136	0.00479
4	sphere	0	0	0	1.11E+03	0	0
5	StepInt	0	0	0	0	0	0
6	Step	0	0	0	1.17E+03	0	0
7	SumSquares	0	0	0	1.48E+02	0	0
8	Trid10	−210	−210	−210	−209.476	−210	−210
9	Trid6	−50	−50	−50	−49.9999	−50	−50
10	Zakharov	0	0	0.00025	0.01336	0	0
11	Bohchevsky1	0	0	0	0	0	0
12	Bohchevsky2	0	0	0	0.06829	0	0
13	Bohchevsky3	0	0	0	0	0	0
14	CamelBack	−1.0316	−1.0316	−1.0316	−1.03163	−1.0316	−1.03163
15	Colville	0	0	0.09297	0.01494	0.04091	0
16	DixonPrice	0	0.66667	0	1.22E+03	0.66667	0.66667
17	Michalewicz10	−9.6602	−4.0072	−9.6602	−9.49683	−9.5912	−8.2894
18	Michalewicz2	−1.8013	−1.5729	−1.8013	−1.8013	−1.8013	−1.8013
19	Michalewicz5	−4.6877	−2.4909	−4.6877	−4.64483	−4.6835	−4.5037
20	Rastrigin	0	43.9771	0	52.9226	11.7167	136.864
21	Rosenbrock	0	15.0886	0.08877	1.96E+05	18.2039	1.46177
22	Schwefel	−12,570	−6909.1	−12,569	−11,593.4	−10,266	−9391.5
23	Akley	0	0.16462	0	14.6718	0	2.1768
24	Griewank	0	0.01739	0	10.6335	0.00148	0.025752
25	Powell	0	0.00011	0.00313	9.70377	2.17E−07	4.4E−08

the results obtained by Karaboga and Akay (2009) for ABC, PSO, GA, and DE. The population size of 50 and maximum number of function evaluation of 500,000 is kept same for all the algorithms. During the iteration, it may be possible that the parameter of newly generated solution cross the search space. In such case the parameter is reset to its nearest boundary value.

From Table 2, it is observed that mean function values obtained by SOPT are better in 3 problems when compared with DE which gives better result in 6 problems. For all other problems both perform equally. When compared with ABC, SOPT gives better results in 4 problems while ABC give better results in 8 problems. SOPT performs better when compared to GA and PSO where it gives better results in 14 and 7 problems respectively. GA and PSO perform better in 4 problems each. Results show that there is no single algorithm that can perform better in all problems.

After improvement in SOPT, it is found that the improved SOPT gives better results than SOPT. In evaluating Dixon-Price, Rosenbrock, Akley, Griewank, Michalewicz10, and Powell functions, improved SOPT is able to obtain exact results while SOPT get trapped in local minima. In all other problems SOPT and improved SOPT give same results. Both version of SOPT is unable to obtain exact results for Rastrigin, and Schwefel functions.

Conclusion

There are number of meta-heuristic algorithms used for the optimization purposes. SOPT is one of the less explored algorithms which consist of two stages, exploration and exploitation stage per iteration. Each stage requires only one function evaluation thus there are only two function evaluations in one iteration. There is only one independent parameter to be set in the algorithm therefore it is easier to experiment with different parameter values for a particular problem. For other algorithms where number of parameters is more, large number of experimentation is required to get best set of parameters value to solve a particular problem. SOPT is compared with some of the most commonly used algorithms ABC, PSO, GA, and DE. Results of experiments show that SOPT performs satisfactorily in obtaining the optimum solution. After improvement in the SOPT algorithm, it is able to obtain exact solutions for 23 problems out of 25 problems. Being a simple algorithm able to get optimum solutions in most of the test problems it can be further explored to solve single and multiobjective problems with constraints.

References

Deb, K., 2004. Optimization for Engineering Design. PHI Learning Pvt. Ltd.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

Hasancebi, O., Azad, K.S., 2012. An efficient metaheuristic algorithm for engineering optimization: SOPT. Int. J. Optim. Civ. Eng. 4 (2), 479—487.

Karaboga, D., Akay, B., 2009. A comparative study of Artificial Bee Colony algorithm. Appl. Math. Comput. 214 (1), 108—132.

Kennedy, J., Eberhart, R.C., 1995. Particle swarm optimization. IEEE Int. Conf. Neural Netw. 4, 1942—1948.